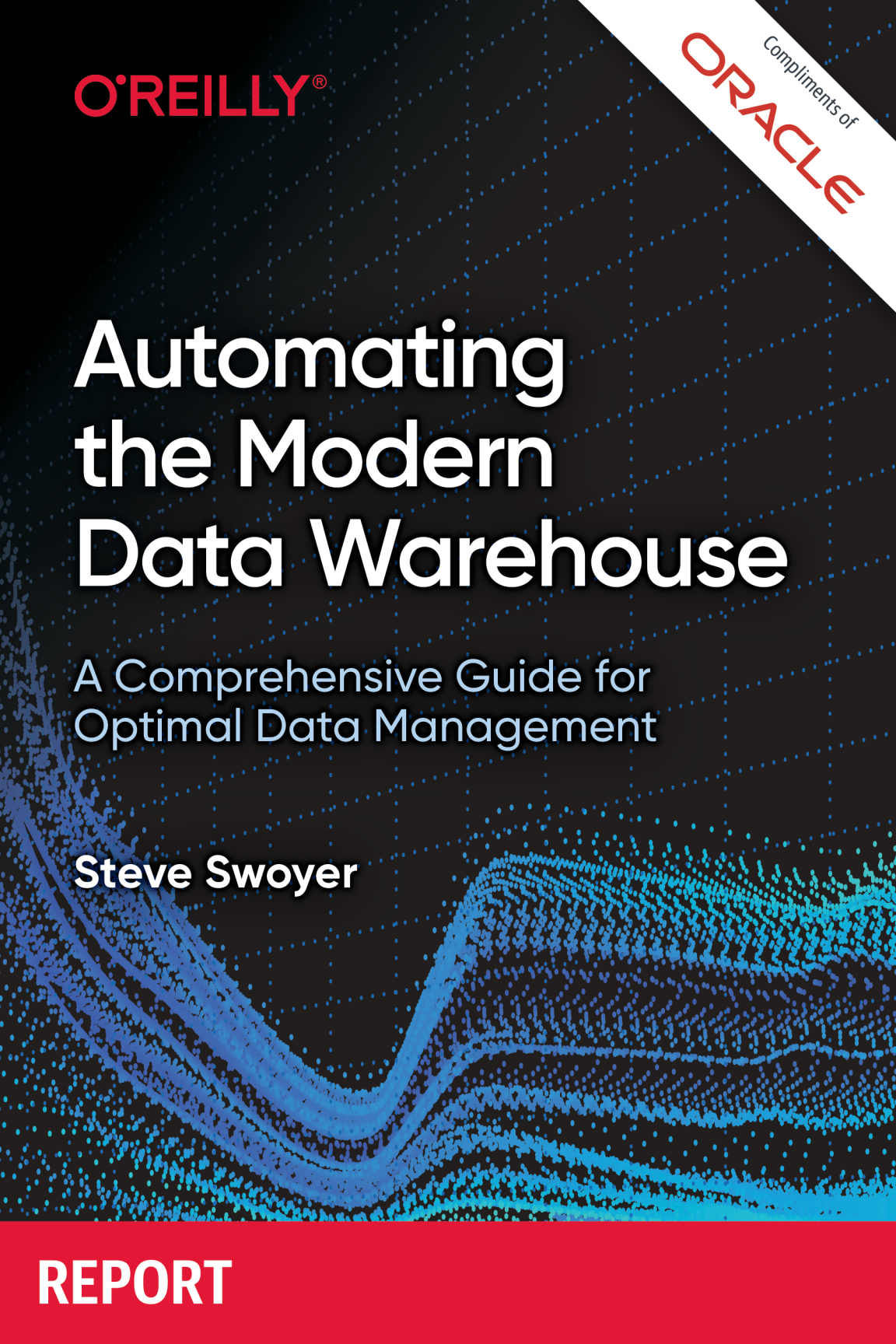
O'REILLY®
Compliments of
ORACLE
Automating the Modern Data Warehouse
A Comprehensive Guide for Optimal Data Management
Steve Swoyer
REPORT

Autonomous Data Warehousing

Delivering data-driven business agility without complexity.

Learn more

oracle.com/database

ORACLE

Automating the Modern Data Warehouse

A Comprehensive Guide for Optimal Data Management

Steve Swoyer

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Automating the Modern Data Warehouse
by Steve Swoyer

Copyright © 2021 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Andy Kwan
Development Editor: Melissa Potter
Production Editor: Christopher Faucher
Copyeditor: nSight, Inc.
Proofreader: nSight, Inc.
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: O'Reilly Media, Inc.

March 2021: First Edition

Revision History for the First Edition
2021-03-04: First Release

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Automating the Modern Data Warehouse*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the author, and do not represent the publisher's views. While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

This work is part of a collaboration between O'Reilly and Oracle. See our statement of editorial independence.

978-1-098-10281-4

[LSI]

Table of Contents

Introduction v

1. Finding Signals in the Midst of Noise 1
The Lives of Analytics 1
Analytics as a Site of Rapid and Ongoing Transformation 2
Diagnosing the Present, Predicting the Future 4
Taking Action, Maximizing Outcomes 5
Putting It All Together 6

2. Modern Data Management 9
Data Management Explained 10
Modern Data Management Explained 12
The Data Lake: A Complement to the Data Warehouse 13
Modern Data Warehouse 15
Modern Data Management and the Cloud 16

3. Machine Learning, AI, and Intelligent Data Management 19
Automation and the Cloud Data Warehouse 20
The PaaS Data Warehouse 21
Using ML-Driven AI to Automate, Improve, and Secure the Cloud Data Warehouse 22
ML and AI Take to (and Take Off in) the Cloud 24
The Multimodel Data Warehouse in the Cloud 25

4. Migrating Your Data Warehouse to the Cloud 29
The Many Modalities of Cloud 30
Cloud Data Warehouse Use Cases 31

The Cloud Data Management Stack 37
Contextualizing Data in the Cloud 40

5. The Data Warehouse in the Cloud. 43
Cloud Data Warehouse Topologies 43
Selecting a Provider 46
Configuring and Managing the Data Warehouse in the Cloud 48
Securing the Data Warehouse in the Cloud 50

Conclusion. 53

Introduction

For the enterprise, cloud is two things: *first*, it's a force for potential business and IT transformation; *second*, it's a mechanism for reducing or controlling costs. Even though enterprises have tended to emphasize the latter at the expense of the former, the most compelling reason to migrate an on-premises data warehouse to the cloud has little to do with reducing costs or taking advantage of cloud's tax-friendly OpEx. In fact, a focus on cost misses what is new and transformative about cloud.

A better, more compelling rationale for cloud migration is to modernize and improve the data warehouse *by substantively automating it*.[1] For one thing, automation of this type—and at this scale—has the potential to eliminate time-consuming, tedious, and rote tasks. For another, it frees up bright, imaginative, creative human technicians—DBAs, ETL developers, business intelligence (BI) developers,

1 The on-premises data warehouse is already a highly automated environment, at least with respect to the scheduling of preinstantiated jobs or tasks; the prioritization (or deprioritization) of workloads on the basis of the privileges of the user or the characteristics of the workload itself; the creation of indexes, metadata, and (in some data warehouse systems) documentation; and so on. But the PaaS data warehouse improves upon this significantly. See Chapter 3 or footnote 2 for more information.

architects, etc.—to focus on more challenging problems.[2] Another benefit of automation at this scale is that it permits IT and the lines of business to respond more rapidly to changing conditions: it becomes possible and cost-effective to accommodate one-off, seasonal, or unprecedented workloads and use cases.

This gets at the *best* rationale for migrating—one that is grounded in the logic of data warehouse modernization itself: to *transform the business*. A focus solely on cost savings sets an organization up to make poor choices in the future. A focus on transforming the business by modernizing the data warehouse, on the other hand, has a conative aspect: it asks, "What should we do differently, better?" This allows companies to better understand their data and improve data driven decision making—boosting innovation, productivity, and efficiency—all while reducing complexity in the organization.

Migration, Modernization, and Transformation

None of these is mutually exclusive. To modernize the data warehouse by migrating to a cloud platform is, ipso facto, to partake of the economic advantages of the cloud. To modernize the data warehouse by migrating to a cloud platform is, ipso facto, to automate at least a share of the tedious, time-consuming, or rote tasks that (with respect to conventional, nonvirtualized data warehouse systems) are usually performed by human technicians. Last, and most important, to modernize the data warehouse by migrating to a cloud platform is, eo ipso, an opportunity to transform the business.

---

2 This is a problem that data warehouse automation (DWA) software has focused on for two decades now. Like DWA tools, PaaS data warehouse services expose a combination of guided, self-service capabilities and ML-powered, rule-driven facilities that automate the exploration, discovery, and profiling of data, as well as simplify (and, if practicable, automate) the creation of data models, business rules, metadata, documentation, etc. But the PaaS data warehouse exposes these features in the context of a fully managed service that, under its covers, automates the allocation and/or deallocation of compute, storage, and network resources, the expansion/contraction of data warehouse capacity, the scheduling and balancing of workloads, the remediation of performance problems, etc. This is the stuff of radical difference: DWA tools were originally conceived and designed for on-premises data warehouse systems that, with few exceptions, lacked the tight integration between hardware and software—to say nothing of the ability to use software to define and abstract hardware resources—that is definitive of cloud infrastructure. The PaaS data warehouse is a completely different animal.

Think about it: today, data warehouse architects, BI developers, and other skilled technicians inevitably squander some fixed proportion of their valuable time servicing the technical debt that encumbers all conventional data warehouse systems. DBAs and ETL developers inevitably squander some fixed proportion of *their* valuable time maintaining the database, data integration, and associated middleware services that constitute the data warehouse system proper. At the same time, an assortment of analytically inclined technicians—among them data scientists, machine learning (ML) engineers, and data engineers—are reliably stymied by the constraints imposed by conventional data warehouse architecture. They cannot get the data they need the way they need it *when* they need it.

The upshot is these and other technicians expend valuable time and energy navigating cruft and manipulating software in order to *manage or access* data instead of *doing stuff with data*.

This Text Is a Guidebook for the Perplexed

The data warehouse in the cloud, and the platform-as-a-service (PaaS) data warehouse especially, has the potential to change this. This book explores the nature and scope of this change, as well as its implications for decision support and SQL-based analytics, to say nothing of its potential impact on adjacent or complementary practices, such as ML engineering and software development.

Along the way, the book explores what is new and different about the data warehouse in the cloud, assessing both the benefits and the challenges—that is, shortcomings, limitations, potential pratfalls, and known edge cases—of transplanting data warehouse architecture into the cloud environment.

Above all, this book delves into the problem of managing coexistence between different types of on- and off-premises data warehouse resources: for example, how to accommodate especially demanding workloads by hosting them in different cloud contexts;[3] how to manage, govern, and knit together data that is distributed across different contexts, including the conventional on-premises

3 Some examples include the on-premises private cloud, the public cloud, the virtual private cloud, the high-performance computing (HPC) public cloud, the regional public cloud, and the on-premises public-private cloud.

data center; how, when, and why to exploit the multimodel and multiengine capabilities that many cloud PaaS data warehouse systems now expose; etc. One of this book's most important themes is not actually broken out into a discrete chapter or section but rather occurs and recurs throughout its pages: namely, that the elasticity that is a defining feature of cloud infrastructure makes it possible for businesses to prioritize the one-off, seasonal, or niche use cases that would otherwise be cost-prohibitive or (a function of both cost and human-resource constraints) impracticable in the on-premises enterprise.

CHAPTER 1

Finding Signals in the Midst of Noise

The data generated by connected devices and other new sources of data has transformed logistics and commerce. It has transformed maintenance of all kinds, in virtually all verticals. It is fueling an ongoing revolution in sales and marketing. It is one of several intersecting factors that have completely transformed data management. To understand the import and ramifications of this transformation, it is helpful to have a sense for what analytics are and how they work. After all, even if we treat data management as an end unto itself, the creation, preservation, and maintenance of data is *always* adjunct to other purposes. Analysis is just *one* of these purposes—albeit one of outsized importance.

The Lives of Analytics

Data is exponentially more useful when it is joined together with other useful units of data to form new combinations. Analytics draws its power from this Lego-like network effect. We create analytics by fusing different units of data into larger combinations called *models*: the star or snowflake schemas that link facts to dimensions in data warehouse architecture are models. Fundamentally, an analytic model is a representation of some slice of the business and its world: for example, sales of Product N in Region X during Period Y to customers with Z_1, Z_2, and Z_3 attributes. In fact, queries like this one are the raison d'être of data warehouse

architecture. The answer to this query already "lives" in the data that populates the warehouse's fact and dimension tables; the data warehouse performs operations that *join* facts to dimensions in real time, creating an analytic model that "answers" the query.

This example also gets at something else: in data warehouse architecture, the role or function of analytics is to *answer* questions. But today's cutting-edge analytic practices invert this arrangement: they seek to *ask* questions. What if Z_1, Z_2, and Z_3 attributes are unknown? What if, in fact, their corresponding dimensions don't even exist in the data warehouse? No conceivable star or snowflake schema can link facts to dimensions that do not yet exist. So the business analyst or data scientist must go to source data—assuming it is available—to answer these and similar questions.

Analytics as a Site of Rapid and Ongoing Transformation

Innovation in analytics is not just a function of fusing Lego-like blocks of data together to create larger ensembles of models. Recent analytic innovation is characterized by the intersection of three distinct trends: first, the capacity to cost-effectively collect, store, and process *more* and *different* types of data; second, the mainstream uptake of ML and especially of advanced ML techniques; and third, the application of this data (of different types and sizes) and of these advanced ML techniques to new problems that involve asking new kinds of questions.

Two decades ago, the data warehouse constituted the analytic center of gravity of the average enterprise: all business-critical data was vectored into it.

It was good to be the king. The warehouse and its constraints dictated the use of a dominant technology to store and manage data: the relational database, or relational database management system (RDBMS). And this dependence on the RDBMS dictated the use of a domain-specific language—SQL—for accessing and manipulating data. These same constraints helped to formalize the use of a set of techniques—starting with extract, transform, and load (ETL)—for engineering the data used to populate the warehouse.

But the data that businesses generate and expect to mine for insights is no longer of a single dominant type (i.e., relational data) and no

longer conforms to a single set of general characteristics. Business data no longer "lives" in the same places (local databases and data warehouses, spreadsheets, network storage, etc.) but is *distributed*, dispersed across on-premises systems, off-premises cloud applications and storage services, mobile devices, the web, and so on.

What is more, business analysts, BI developers, and DBAs are no longer the only people who work with data, nor is BI itself the only —or even the *primary*—practice area for data work. BI and data warehouse people compete with data scientists, data engineers, ML engineers, and other specialists for access to more data, to fresher data, and to data of different types.

This gets at another major change—one that has to do with the role human intelligence now assumes in both the analysis of data and the production of analytics. The upshot is that human intelligence now allocates a large (and growing) proportion of analysis to *machine* intelligence, which makes it possible to automate not only the task of analysis itself but that of decision making with data preparation, data enrichment, and virtualization of results.

Human intelligence is likewise training machine intelligence to *replicate* itself—that is, to produce its own analytic models. (This is the focus of ML in general and especially of deep learning, reinforcement learning, and other advanced ML techniques.) The growth of ML is a reprise of a familiar story arc. Until relatively recently, and with a few notable exceptions, human "computers," not machines, performed most mathematical calculations. As the complexity of the calculations involved and (especially) the scale at which they needed to be calculated increased, machine computers at first complemented and then ultimately replaced human computers. The same is happening with analytics, whereby organizations are replacing human analysts with automated analytic technologies and human-directed analysis with machine-directed analytics. In the present, the bulk of analysis is already performed by machines; in the future, almost all analytics will be produced by machines, too.

Analytic practices are also changing. The BI practice area is now complemented by new practice areas such as data science and ML/artificial intelligence (AI) development. The people and machines who work with data no longer expect to use a single means of access —an ODBC interface—and a single common language (SQL) to access, manipulate, and query data. And analytics as such is no

longer the remit of a single practice area or a single domain: the data warehouse and BI; data science and its products; ML engineering and its products, etc. Rather, almost all applications and services will incorporate analytic capabilities, with the result that the consumption of analytics will, in a sense, become commoditized.

Lastly, the batch ingest model that was ideal for the data warehouse is unsuitable for emerging data warehouse use cases, to say nothing of data science, ML/AI engineering, and other analytic practices. Real-time data warehousing is not in any sense new, of course; what *is* new, however, is the expectation that data should be as fresh, as close to real time as possible. The upshot is that data must now be ingested as it arrives: as it pulses, as it streams; as it trickles, dribbles, or deluges.

Diagnosing the Present, Predicting the Future

Most BI work consists of combining customer, product, sales, and similar data into multidimensional views. The warehouse is still *the* killer app for asking questions of this kind. But access to data of diverse shapes and sizes permits businesses to ask new, different, more ambitious questions—questions that involve discovering as-yet-unknown relationships between bits and pieces of data.

Consider the twenty-first-century cargo ship. Like other modes of commercial transport—railcars, tractor trailers, and aircraft—the cargo ship now bristles with sensors of different types: temperature sensors; sensors that record the frequency and impact of bumps or jostles; sensors that measure motion; sensors that detect chemicals and gases, such as those correlated with cargo spoilage. These sensors generate enormous volumes of data, a small subset of which gets transmitted back to the shipping company, sometimes in real time. This data is a potential treasure trove for business.

Raw sensor data is of limited use in data warehouse-driven analytic development, where modelers and business analysts construct analytic views grounded in *known* relationships in available data. But the data generated by sensors lets an organization ask questions that have a definitive *inductive* quality: they're attempts to reason backward from effects to causes, attempts to discover *unknown* relationships that permit businesses to diagnose problems in the present, attempts to make predictions about the future and to take action. For example, in a ship carrying, say, bananas and mangos from

Puerto Quetzal, Guatemala, to Seattle, Washington, is there a possible relationship between prolonged jostling or bumping during loading, transport, and unloading and higher rates of spoilage? In other words, is it possible to correlate the frequency and severity of impact with the rate of fruit spoilage? What about air quality—or, more precisely, what about the mix of gases in the air in the shipping containers that house the cargo of produce and mangos? Can a *combination* of these and other factors be correlated with spoilage? Also, is it possible to detect early warnings of spoilage—for example, in the presence of certain noxious chemicals or gases? If so, what could a shipping company do to prevent this?

Taking Action, Maximizing Outcomes

Answering these questions depends on the availability and mainstream uptake of advanced statistical tools and techniques. These are questions that involve modeling a finite set of interactions among known variables in a slice of the world—namely, a shipping container stowed in a cargo ship traversing a weeks' long voyage from a tropical to a subtropical climate—in which not all variables can be anticipated, let alone modeled. Data scientists and other skilled people use statistical techniques to determine, first, if a question has a statistically significant "answer" and, second, how to interpret—how to *use*—this "answer." The strongest "answers" are usually products of combinations of different factors. So bumps and impacts alone aren't strongly correlated with higher-than-average rates of spoilage. However, in combination with other factors (e.g., modest fluctuations in temperature, anomalously high proportions of certain gases in the atmosphere), the case for a correlation seems much stronger. Once the data science team decides that these relationships are statistically significant, it refocuses its efforts on another problem: what can the business do with this knowledge?

This is one potential use for AI engineering. Not "AI" in the mode of artificial *general* intelligence, which is analogous to human cognition in its self-reflective dimension. AI engineering is the application of ML to identify and diagnose problems; the use of automated rules engines to trigger interventions on the bases of a diagnosis; and, finally, constant monitoring by an AI feedback loop to measure the effectiveness of interventions and to self-correct if necessary.

Businesses want to use AI to increase productivity, to accelerate processes, to optimize outcomes, to forestall reversals, and, moreover, to deliver wholly new products and services. The cost-economics and the colocality of ML, AI, data science, and other services in the cloud are ideal for the AI engineering use case: cloud's primary storage layer is inexpensive and (from a subscriber's point of view) practically unlimited; cloud's elastic character permits an organization to grow or reduce the storage, compute, and network resources it consumes; the availability of tightly integrated cloud development services—which in most cases also accommodate the tools, libraries, and techniques that data scientists, ML engineers, data engineers, and other technicians prefer to use—is still another selling point, as is the emergence of full-featured, cloud-centered data integration services.

And because so much data originates in the cloud, the cloud is a logical locus of data ingestion and integration. The cloud comprises a highly integrated—and, in the case of homogeneous (vendor-specific) deployments, *vertically* integrated—site for data science and AI engineering.

Putting It All Together

This is a world in which the data warehouse has an increasingly vital role to play: not as a kind of subaltern to otherwise dominant data science and AI engineering practices, but rather as the privileged destination for the analytic products—the insights—of data scientific and ML development.

Think about the twenty-first-century cargo ship, bristling with sensors amidst its Tetris-like assortment of shipping containers. Assume that its onboard sensors record a higher-than-usual number of severe impacts during loading. Assume, too, that sensors record variations in temperature and the presence of a specific gas (say, ethylene) in several onboard containers. Given the correlation between each of these factors and a definite rate of fruit spoilage, what does this mean for the projected value of the ship's cargo? This is, of course, a question that the data scientist could answer; it is also, however, just the kind of question that a business decision maker might want to ask *on a recurring basis*. And so the data science team, working in tandem with a business analyst, adds a new dimension to the warehouse to incorporate this analytic "fact." The upshot is that

businesspeople can now pose questions and make projections about spoilage based on events that (just five years ago) could be modeled only imprecisely. For example, what if the ship were to dock at the Port of Los Angeles? Would getting the mangos and bananas to market earlier significantly arrest the rate of spoilage?

This is the vital role of the data warehouse in the context of modern data management and a cluster of complementary analytic practices. Owing to a combination of factors, the focus of these practices—and of data management itself—has shifted to the cloud. This is the biggest change by far. Now as ever, the data warehouse in the cloud is positioned as the go-to engine for day-to-day and strategic business decision making. It is no longer the center of its own planetary system, however.

CHAPTER 2
Modern Data Management

Data management is not an end unto itself. That is, the management of data is always adjunct to some other purpose, some other goal, some other use. One of the most important uses of data is in support of business decision making, but decision making is neither a monolithic domain—that is, a problem area in which the stakes, priorities, requirements, and expectations are uniform or reproducible across all relevant instances and applications—nor is it the sole domain in which data gets used. For example, data is generated and consumed by operational applications that span business processes and that support day-to-day business activities. This type of usage has less to do with decision making than with ensuring the uninterrupted and (with respect to core business processes and workflows) unmediated operation of the business itself. In many if not most cases, this use case does not have an analytic component: data is generated by an application or service, written to a database or serialized in a data structure, requested by and exchanged with other applications and services, and so on.

Another important use for data has to do with monitoring—or, in emerging software architectures, *observing*—the state and performance of the business itself, along with its constitutive processes, IT systems, and other assets. This use case is usually married to a related task—that of analyzing and interpreting problems or anomalies. Sometimes decision making attends the analysis and interpretation of a problem, sometimes not. But the material fact is that decision support, while an especially critical data use case, is nevertheless just *one* use case among many.

With this in mind, it is worth looking at what data management *does* in order to get a sense for what needs to change—not only with respect to the data warehouse in the cloud, but with respect to data management in a hybrid environment that spans both the on-premises enterprise *and* the cloud.

Data Management Explained

To grasp the "why?" of data management, it is helpful to know a little bit about a concept in law called "riparian rights." In Anglo-American law, riparian rights has to do with the conventions, statutes, and precedents that govern the use of land that adjoins a river—or, notionally, any body of water that has banks or a shoreline. The basic idea is that a person who owns property that lies adjacent to, say, a large oxbow lake enjoys certain rights with respect to the reasonable use of the water that circulates in that lake. Traditionally, the extent of a landowner's use might be determined by the proportion of her property that directly abuts the river, by the size and/or duration of her holdings, etc.

So far, so good. But the word "reasonable" does a good deal of heavy lifting in this context. In the United States, riparian law distinguishes between "natural" (i.e., domestic uses) and other types of uses—for example, entrepreneurial or industrial uses.[1] People need clean, fresh water to drink, to cook with—that is, to survive. People (and municipalities) need clean, fresh water for the purposes of sanitation. To this end, riparian law privileges natural use as "reasonable" over and against other types of use; in other words, if industrial usage is in some way inimical to natural usage, it is, ipso facto, unreasonable.

The analogy to data and data management is fairly obvious, even if the analogy itself ultimately breaks down. Think of data management as a kind of riparian rights regime for data. Its raison d'être is not just to define and enforce rules with respect to how data gets used, but to prioritize certain kinds of uses, which are analogous to "natural" uses in the logic of riparian rights, over and against others, analogous to industrial or entrepreneurial uses. For a long time, data management's primary reason for being was to ensure that a class of

1 Ludwik A. Teclaff, "What You Have Always Wanted to Know About Riparian Rights, but Were Afraid to Ask," *Natural Resources Journal* 30, no. 1 (Winter 1972): 41.

privileged downstream uses—operational reporting, ad hoc query, and certain kinds of advanced analytics—was fed a reliable stream of cleansed, consistent, high-quality data.

In a nutshell, data management's role is to define and enforce *constraints* with respect to the production and use of data. To this end, it defines policies, processes, and controls that govern the production of cleansed and consistent data, and it ensures that the data so produced comports with predetermined quality thresholds. Data management's traditional focus was the data warehouse, which itself defines a conceptual architecture for storing, managing, and processing data that is useful for decision making. In traditional data management—which is to say, traditional data warehouse architecture—data management policies, processes, and controls produce high-quality data that gets stored in, managed, and processed by the data warehouse, chiefly to support day-to-day business decision making, as well as business forecasting and strategic business planning. In this sense, what is meant by data management has long been inseparable from what is meant by decision support.

The problem is that the scope of decision support has changed radically since data warehouse architecture was first codified in the late 1980s. Chapter 1 discussed one of the biggest of these changes, exploring how data scientists, ML engineers, and other skilled practitioners expect to work with data that is derived from a diversity of sources—much of it nonrelational data. Another major change (explored in detail in Chapter 1) has to do with the practices, tools, and techniques that people (and machines) use to engineer data and to produce the analytics that support, inform, and (increasingly) *automate* decision making. And a third significant change has to do with the movement of data and analytic development from the data warehouse to new platforms.

The upshot is threefold: in the first place, the data warehouse and especially data warehouse-driven analytics are no longer the sole focus of enterprise data management; in the second place, data management has had to adapt to support different types of analytic practices, along with different types of analytic creators; and, in the third place, data management must reckon with two problems—*data distributedness* and *data heterogeneity*—that are at once characteristic of the way data is created and used (by humans and machines alike) and that have been exacerbated by the same social, economic, and

technological transformations that contributed to the emergence of the cloud.

Modern Data Management Explained

In classic data warehouse architecture, the ETL layer was the locus of data ingestion and integration. ETL processes conditioned data for a single privileged analytic target: the data warehouse. One consequence of this was that different types of analytic practices tended to cluster around the warehouse; another was that the warehouse and its data integration processes constituted the primary focus areas of enterprise data quality assurance, data governance, and information security assurance.

The data warehouse has always been a multipurpose platform. It is optimized for storing, managing, and retrieving relational data. It is also a data-processing engine, optimized for performing operations on relational data. The RDBMS engine at the heart of the warehouse is likewise adept at ingesting, performing operations on, and retrieving *tabular* data. This encompasses the telemetry data that is generated by sensors; the logs generated by embedded devices; data extracted from hierarchical databases, network databases, and key-value stores; data serialized in CSV or similar file formats, etc. It is usually fairly simple to engineer this data for use with the data warehouse and its core RDBMS engine. The catch is that an RDBMS expects to apply a schema to all of the data that it ingests: data gets mapped directly to rows and columns in accordance with a predefined data model. Extraneous data—data that does not map to the warehouse's data model—gets discarded. This schema-on-write model is ideal when the structure of the data and the applications for which it is to be used are known in advance. But schema-on-write is not always sufficient, much less ideal, when the opposite is true. This is why data scientists, ML engineers, software developers, and other advanced users prefer to work with source data *as it is*—with raw source data, not just with extracted values, pairs, etc. These practitioners cannot anticipate just what data (or what values encoded in the data) will be useful.

Beginning about 15 years ago, a new type of data store, the NoSQL data store, emerged to address this problem. NoSQL employs a schema-on-read approach in which the data store behaves much like a file system, ingesting objects—CSV, XML, and TXT files; audio,

image, and video files; and so on—and storing them in their native formats. Users can either retrieve these files as they are or (using the data store's built-in data processing engine) apply a schema to them when they access them. Since then, a refinement of this vision—the data lake—has emerged to complement (and, to a degree, assimilate) the NoSQL use case. The data lake consists of a data management layer superimposed over a cheap cloud object storage layer—basically, a practically inexhaustible distributed file system.

The emergence of NoSQL created a kind of cleavage in data management, putting the SQL and NoSQL camps at loggerheads with one another. Over time, however, these capabilities converged into a single logical context: what market-watcher Gartner calls the *logical data warehouse*. Before we explore what this is, let's take a look at the data lake to see how it compares with the data warehouse.

The Data Lake: A Complement to the Data Warehouse

What is useful about the data lake is that it can store data of any size or type. Not just relational data, but CSV files, log files, XML files, and JavaScript Object Notation (JSON) files, along with "multistructured" objects (e.g., music, video, or image files) that do not expose predefined data models.

The data lake is in fact an ideal repository for *file* data, content for which the RDBMS is not always ideal. In the cloud, most data lakes are layered over *object storage*, which (for all intents and purposes) describes a virtual, distributed file system, albeit one that stores metadata about different types of objects. The data lake exposes a (more or less) rich metadata layer and, optionally, a data catalog facility of some kind. Under its covers, the lake incorporates features that aim to simplify—and, where practicable, to automate—data access, data preparation, and data movement for different types of users and requestors. With this in mind, it is useful to note a few things about the data lake:

1. It is conceived as a practically unlimited repository for data of all types and sizes. In the cloud, especially, each and every data lake is potentially the largest and deepest in the world (i.e., by volume of data). This is why the site of the data lake quickly

shifted from the on-premises enterprise to the cloud. Storage in the cloud is much cheaper and practically limitless.

2. It supports a wide variety of analytic practices and a breadth of different analytic use cases. The lake is an ingest point for data from operational systems. If necessary, it can exchange data with the cloud data warehouse and with the on-premises warehouse, too. As a cost-effective, practically unlimited storage repository for data of all types and sizes, the data lake is a popular ingest point for the telemetry data that streams from sensors and mobile devices; for the log files that are generated by embedded devices, control systems, and similar signalers; and for the messages—encapsulated as JSON objects, etc.—exchanged by applications and services. The lake exchanges data with a medley of other cloud services, too.
3. Each analytic practice and each use case expects to do different things with data. Business analysts are mostly interested in relational data sourced from operational systems—many of which now live in the cloud. They prefer to have this data as raw extracts, and they expect to be able to model it and transform it according to their own requirements. Data scientists and ML engineers work with relational, semistructured, and polystructured data. They need to be able to prepare and transform this data for different kinds of use cases. The lake gives them a context in which to load the data they need and to perform operations on it.

The data lake's most distinctive characteristic is a not-so-flattering one: namely, almost everything about it entails compromise. The cloud data lake's practically unlimited storage is useful, to be sure, but it comes at the expense of critical performance criteria—namely, high latency and inconsistent data transfers relative to on-premises storage—that are death to many analytic workloads. And the data lake's versatility with respect to storing data of different types and sizes? The reverse of this is that—unlike a relational, graph, or document database—the lake is not tuned for storing or retrieving *any one specific* type of data. As for the lake's versatile data processing features, in most cases, for most workloads, these are no match for an RDBMS, a graph database, or other best-in-class engines.

Modern Data Warehouse

Modern data management distinguishes between two primary contexts for storing, processing, and working with the data that is grist for analytic development: the data warehouse and the data lake.

Data management does not treat these as two separate but equal contexts but rather as a logical, if federated, whole—as a brain with two lobes, so to speak. One lobe, the data warehouse, remains the primary focus of decision support: it is the system of record, the engine that powers operational and production reporting, ad hoc query, and many types of advanced SQL analytics. Now as ever, the data warehouse is the primary support for day-to-day business decision making and for forecasting and strategic planning. It is the foundation of business performance monitoring. The other lobe, the data lake, is home to all of the multistructured data that is of potential use to the business. Because it combines inexpensive storage with several different types of built-in data processing engines, the data lake hosts a diversity of analytic practices, too—including practices that work with relational data.

This invites obvious questions, however: How does data management expect to unify these otherwise disconnected contexts? How is it possible to manage data—to enforce reasonable constraints—in two different contexts, each of which has its own requirements and expectations?

Enter the *logical data warehouse*, a virtual abstraction layer that aims to knit together potentially useful data sources and processing engines—regardless of where they are located, the kinds of interfaces they expose, the specifics of the underlying systems on which they are hosted, etc. The logical data warehouse does not alter or disrupt the data warehouse, the data lake, or—just as important—any of the data sources it knits together. Rather, it describes a model in which an organization constructs *virtual views* of data that is distributed between and among disparate sources. Potential data sources include cloud and web services, RESTful or otherwise, legacy sources (such as mainframes), along with other RDBMSs and operational applications. So, for example, a data modeler or business analyst could construct a virtual view that exposes VSAM data stored on a mainframe in the same context as data that lives in the SaaS (software-as-a-service) or PaaS cloud. That is the vision, in any case.

However, this vision elides a staggering degree of behind-the-scenes complexity. Its key enabling technologies include data virtualization and data catalog services, along with complementary technologies, such as graph databases. (See Chapter 4 for more on these technologies.) Under the covers, the logical data warehouse uses data virtualization to expose unified views of distributed data, contextualizing data that lives in the warehouse or data lake with (for example) data that is hosted in far-flung cloud services. It exposes data catalog services that permit knowledgeable users (e.g., IT people, developers, business analysts, data scientists, ML/AI engineers, etc.) to search for and discover useful data that has not yet been exposed via data virtualization. And it makes use of other, better-known technologies (e.g., caching, data synchronization, and data replication) to accelerate common queries or facilitate access to frequently used data.

The logical data warehouse is a pragmatic application of a kind of sociotechnical federalism. It is a means of consolidating and governing information with a light hand: a logical overlay in which virtual data structures are used to represent source data (and in which transformation logic is applied to these virtual data structures) *without physically altering source data*. More important, the logical data warehouse model, or something like it, also permits the people who manage and use the data warehouse or the data lake—or any other repository or data source, for that matter—to enforce their own standards and policies with respect to the quality of data, or to the priority given to certain users, use cases, or workloads as against others, and so on. In this way, the logical data warehouse affords visibility into and facilitates access to data that is not only physically dispersed but disconnected—walled off—by virtue of policies, procedures, and other impedimenta.

Modern Data Management and the Cloud

Once the data warehouse joins the data lake in the cloud, the focus of data management shifts to that context as well. The good news is that managing data in the cloud is not radically different from managing data in the on-premises enterprise. This claim presupposes an important qualification, however: namely, that the enterprise data management practice has reconciled itself to the necessity of managing data *in accordance with its destined uses*. The classic data warehouse is still associated with a rigid, inflexible data management

regime that gave priority to core BI reporting and analytics at the expense of other practices—most of which prefer to work with data that is less cleansed, less consistent, and less conditioned (i.e., of a lower "quality") than the data populating the warehouse.

The practice of data management must be flexible enough to manage and govern data according to its destined uses. In the same way, the problem of governing data in the cloud does not necessarily demand new processes, policies, tools, roles, and so forth. This is also true of basically all data quality standards, processes, and tools. It is less a question of *extending* these artifacts to the cloud—remember, most of them were developed with the constraints of the on-premises environment in mind—than of *updating* them to accommodate what is different about the cloud.

For example, in the cloud, as in the on-premises environment, organizations have a responsibility to safeguard the integrity and consistency of their applications and data. In both the cloud and the on-premises environment, organizations need to be careful about how they store sensitive data. They must also comply with applicable (country-, political union-, state- or region-specific) data privacy, data retention, and data deletion regulations. The good news is that the cloud data warehouse—and the PaaS data warehouse, especially—has the potential to simplify these tasks. For one thing, a standard cloud practice is to isolate sensitive data from nonsensitive data; for another, most cloud PaaS data warehouse platforms encrypt in-flight data by default. (Both practices are inconsistently applied in the on-premises enterprise.) Similarly, some cloud data warehouse and data integration services incorporate built-in mechanisms designed to make it easier for organizations to define and enforce data governance policies in connection with data access, movement, or deletion. Some cloud providers enforce these policies by default, for example, by preventing requestors from outside of a political union from accessing sensitive data that is governed by the statutes or regulations of that political union.

CHAPTER 3

Machine Learning, AI, and Intelligent Data Management

At one time or another, most of us who have used cloud services have marveled at the way they elide the underlying complexity of different problems and tasks. For the most part, this is a function not just of automation but, more specifically, of *rule-driven* automation. This kind of "smart," rule-driven AI is not distinctive to cloud as such, and it is not necessarily new; what *is* different is the unprecedented scale of the cloud—not with respect to the number and capacity of its virtual hardware resources, but rather with respect to the number and variety of machines that live in it and people who use it.

The coinciding of people and machines in a single context lends itself to the purpose of *studying* them: that is, collecting information about and analyzing their behaviors at an unprecedented scale. This advantage *is* distinctive to cloud. It gives cloud providers a potentially massive dataset for training ML models and designing "AI" functions: that is, software "robots" that automate actions when they detect problems or events. These software "robots" are used to automate a range of tasks, both common and esoteric. Over time, then, cloud providers have automated a growing share of tasks and remediations.

This is radically different. There is nothing like it in the on-premises data center. Yes, much of this cloud-centered automation ultimately makes its way to the on-premises enterprise. But the automation

gets implemented in the cloud first. This is why the cloud is already the locus of innovation in software development. This is as true of the cloud data warehouse as it is of any other cloud service.

Automation and the Cloud Data Warehouse

The on-premises data warehouse is already a highly automated environment.

The ETL or ELT processes that feed data to it are scheduled, controlled, and audited. They generate logs that not only record essential data lineage information but automatically trigger alerts if something goes wrong. The warehouse's backup processes are automated, as are its recovery processes. Just as important, the warehouse's *internals* are highly automated, too.

In this regard, the database at the core of the warehouse is a marvel of automation. The best on-premises warehouses manage workloads at more or less granular levels, such that each workload gets allocated compute resources in accordance with its importance or, alternately, with the role of its initiating user. Certain workloads run in the background, using resources as they become available, while other workloads expect to run as quickly as possible with as many resources as possible. In MPP data warehouse systems, the core database automatically manages parallelism and concurrency: breaking up a single workload and distributing it across multiple nodes for parallel processing, scheduling multiple workloads to run concurrently across multiple nodes, managing job dependencies between workloads, etc. (Some cloud services are better at this than others, however; workload management is inevitably a function of the maturity of the underlying database.)

Optionally, most warehouses automatically manage multitemperature storage, such that different kinds of data get treated differently—and, optionally, stored in different contexts. Frequently accessed data lives in "hot" or fast storage—or is permanently pinned in in-memory cache—while less frequently accessed data lives in the "warm" tier: sometimes local, direct-attached storage; sometimes in a SAN; sometimes in local object storage. Data that is rarely accessed ("cold") might live in a network-attached storage or in a (local or cloud) object storage layer. These are just a few examples of automation in the context of the on-premises data warehouse. Essentially all of this is replicated in the IaaS (infrastructure-as-a-service) data

warehouse—with the bonus that the IaaS environment in which the data warehouse lives is itself highly automated. The IaaS provider exposes features that simplify and automate the processes of expanding (or reducing) the capacity of a system, recreating or mirroring a system, and so on.

The PaaS Data Warehouse

The PaaS data warehouse in the cloud ups the ante with respect to automation, however. It retains these advantages but also benefits from a kind of cloud-specific vertical integration.

For example, the PaaS data warehouse exposes GUI-based wizards and guided self-service features, the aim of which is to automate—or substantively to accelerate—many of the tasks that tend to consume a disproportionate amount of time or which entail nontrivial labor in the on-premises environment. These apply not only to the initial creation, setup, and configuration of the cloud data warehouse, but to a subset of the tasks usually associated with the ongoing management and maintenance of the warehouse itself, such as adding extra storage, compute, or network resources; adding new tables and optimizing table spaces; creating indexes to simplify access or to improve performance; and redistributing data to avoid skew, which is primarily a problem in connection with MPP data warehouses and with certain kinds of sharded configurations. In the same way, most PaaS services expose guided, self-service data exploration, data discovery, and data preparation features. On the one hand, these features aim to reduce the time it takes to provision data for the warehouse. For example, stakeholders are able to work with business analysts to identify useful data sources and determine how this data is to be transformed. On the other hand, they also simplify the process of scheduling data extracts as recurring or repeatable data flows. The cloud-based data integration environment automatically tracks and generates lineage information along with metadata, too.

Recent innovations in the on-premises data warehouse attempt to replicate these features. This suggests that enterprise customers now expect cloud-like usability—even in the on-premises data center. However, the data warehouse-as-a-service boasts automation advantages that cannot easily be matched, let alone surpassed, by the conventional, on-premises warehouse.

Using ML-Driven AI to Automate, Improve, and Secure the Cloud Data Warehouse

The PaaS data warehouse uses automation of another kind: namely, ML-powered, rule-driven automation—automation that monitors performance, automation that detects issues and anomalies, and automation that triggers remediations. "AI" in this context refers to a kind of function-specific machine intelligence: that is, AI software robots that are designed for specific purposes, AI robots that do specific things. The PaaS data warehouse is not Skynet: it is not self-aware.

Instead, the PaaS provider trains its software robots to recognize common or recurring conditions, along with (as they are identified) edge cases. For example, a provider might develop software robots that (a) detect the combination of conditions associated with hardware failure (e.g., degraded performance and/or complete unavailability) and (b) trigger one or more predetermined steps based on a set of predefined rules. This focus on hardware as a source of problems elides an important fact: with few exceptions, the PaaS data warehouse provider does not own and operate its *own* cloud infrastructure. Instead, it contracts with one or more public cloud computing infrastructure services to host its platform. From the provider's perspective, the problem is at least one level of abstraction higher: rather than an issue of identifying and diagnosing, say, impending hardware failure—hardware is the upstream service provider's problem—it is one of determining and, if possible, offsetting the impact of hardware failure *on the delivery of their service*. Thanks to the colossal scale of the cloud, most such problems can be solved by provisioning extra capacity and by rearchitecting to route around underlying problems. Ideally, this happens automatically, without the subscriber experiencing downtime or service inconsistency. Their software robots watch, detect anomalies, and trigger responses.

AI software robots cannot address all possible issues, of course. Again, however, the PaaS provider has a built-in advantage: it hosts dozens, hundreds, perhaps *thousands* of subscribers. It collects data that it uses to train the ML models that power its function-specific AI robots. This makes it possible for PaaS providers to diagnose common and recurring issues and to identify a large number of edge cases, too. With enough data, ML engineers can train models that

detect problems or anomalies—and, moreover, design software robots that trigger predefined actions. From the outside, this looks like so much AI magic; in the background, it involves the practical application of ML to solve problems.

The effects of AI-driven automation in the PaaS data warehouse are not confined exclusively to detecting problems. Most PaaS providers monitor and collect information about how their subscribers use their platforms: the tasks that DBAs perform, the indexes they create (e.g., What kind of index is it? *Why* did they create it? That is, adjunct to what purpose? Is this use case generalizable?), as well as the kinds of queries that users run, how these queries perform, and so on. The priority is to identify generalizable actions that providers can train AI robots to perform. All software vendors use feedback from their customers to improve their products, but the PaaS cloud has the potential not only to increase the capacity for improvement but to accelerate the rate at which improvements are delivered.

As defined in this chapter, "AI" also has implications for securing the data warehouse in the cloud. Again, PaaS (and, in this case, IaaS) providers have the capacity to collect a massive amount of data about how people and machines tend to behave in the context of their services: who or what accesses the service, what do they do with it, and, crucially, what do they *attempt* to do with it. On this basis, ML engineers can train ML that identifies different kinds of possible attack scenarios: for example, a user is attempting to access data they (or it) do not typically use; a user (or machine) tries, and fails, to access data; a user (or machine) initiates several bulk extracts. Once their models are trained, ML and AI engineers can create different kinds of software robots designed to perform predefined actions.

The usefulness of ML-powered AI in the cloud gets at something else. Again, AI is just another name for a kind of ML-powered, rule-driven automation. The data warehouse itself is already a good example of smart, rule-driven automation. However, for the reasons already stated, transplanting the data warehouse *into* the cloud makes it the beneficiary of ML-powered AI on a scale that is unimaginable in the on-premises data center. This is no less true of the data lake. With respect to the data lake, too, ML-powered AI is used to automate common (sometimes esoteric) tasks, detect problems and anomalies, secure data against potential attack, etc. Automation is more versatile in this context.

ML and AI Take to (and Take Off in) the Cloud

The analytic practices of today are a lot more specialized. The term "practices" is significant in this context. Twenty years ago, analytics was a comparatively monolithic practice area: data science did not really exist, ML was a niche area, and even though almost all large organizations employed statisticians and owned licenses for commercial statistical analysis software, the most common analytical practices (a) still worked with relational data and (b) focused on (or were focused by) the data warehouse. You just could not point to the diversity of roles (e.g., data scientist, ML engineer, data engineer, AI engineer, etc.) that is common today. Cutting-edge use cases in the mid-1990s and early-2000s involved the integration of data from geographical information systems (GIS) with customer, product, and spatial data in the warehouse. (The retail vertical was out in front of this.) Data mining was commonly used, and organizations in some verticals (finance, retail) experimented with statistical methods to predict fraud and customer churn. Similarly, rudimentary work on AI—e.g., engines that were used to power rule-driven automation—was likewise becoming common. Organizations were beginning to combine the insights of predictive analytics (i.e., the application of ML functions to data problems) with rule-driven automation to monitor the behaviors of their customers and to trigger actions on the basis of fraud, likely customer churn, or the unavailability of certain products. (The retail sector, again, was out in front with this class of "next-best-offer" analytics.) By the early 2000s, predictive-analytic use cases had become common enough that GIS and certain kinds of ML functions started appearing in the database engines at the core of the data warehouse itself.

Until very recently, then, the data warehouse and its tools effectively circumscribed the domain of analytics in the enterprise. Analytics consumed relational data that lived in the warehouse—or which was discarded by the ETL processes that prepared data for use with the warehouse. Analytic practices focused on and, in most cases, were hosted by the data warehouse, too. As a general rule, analytic roles were also less diverse, sorting into two buckets: business analysts and statisticians. There *was* a role for self-service, but it mostly focused on spreadsheets; it was unlike today's BI discovery practice.

This is no longer the case. Analytics has diversified. Concomitant with this, the domain of enterprise analytics has escaped or outstrip-

ped the data warehouse—not in the sense that analytic practices have abandoned the warehouse or left it behind, but rather that the data warehouse has a somewhat different role in (and relation to) data science, ML and AI engineering, and data engineering. For example, a data scientist works with the warehouse and its data; they work *on*—that is, in the context of—the warehouse and its data; but they tend to do most of their work in *other* contexts, with *other* types of data (in the data lake, for example, which is home to *all* of the data they require for their work). They can identify useful data and invoke the data lake's built-in data processing features to prepare it and extract it to the workspace of their choice: an analytic sandbox in the data lake itself, a cloud data warehouse service, a cloud analytics service, etc. In the on-premises environment, the data scientist was constrained by practical—and sometimes show-stopping—physical constraints, starting with the available capacity of the data warehouse itself. In the cloud, they have considerably more freedom.

The Multimodel Data Warehouse in the Cloud

In addition to a core RDBMS engine, some data warehouse systems incorporate other engines that permit them to perform different types of operations on data stored in the warehouse context. With respect to the cloud, for example, several PaaS data warehouse services expose graph, spatial, and time-series data processing capabilities. Most expose Python, R, and/or Java interfaces, which permit them to perform ML processing, too; however, several support processing relational data against ML models *in the context of the database kernel*. Some ingest and store this data as relational data, translating between data models as needed. Some create dedicated data types—and/or in-database facilities—to store nonrelational data. (Time-series data, for example, is relatively simple to ingest and manage.) The upshot is that the modern data warehouse constitutes a single, versatile platform for storing, managing, retrieving, and processing data of different types. Versatility of this kind is valuable for several reasons, not least because it permits users and machines to query graph, spatial, and time-series data using SQL (as well as Python, R, and Java) *where it lives*—that is, in the warehouse context—without moving it to an external engine for data processing. This is hugely convenient. In practice, a multimodel data

warehouse can accelerate data processing for many common workloads (see Figure 3-1).

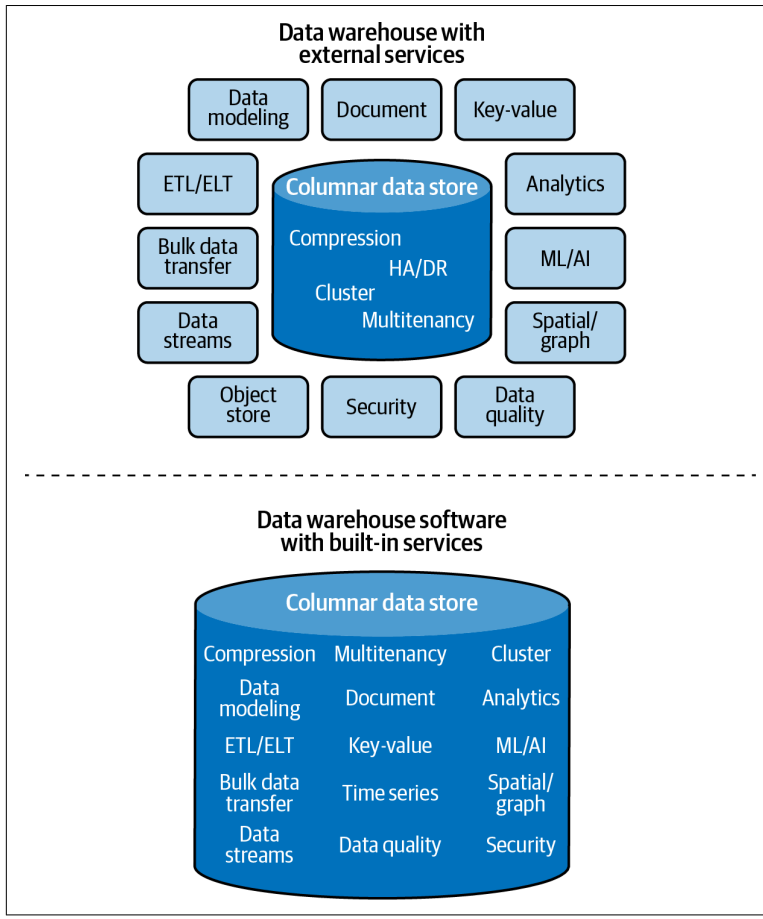

Figure 3-1. Built-in versus open and assembled data warehouse

For example, a large retailer might opt to run sales, demographic, and other kinds of data against thousands of ML models *in the RDBMS itself*. The ML models themselves might be relatively simple, but the compute power required to process data against thousands of models at once is significant. An alternative would be for the retailer to instantiate batch processes that extract this data from the warehouse, load it into an intermediate repository, transform and tailor it to the needs of the ML models, and process the data against the requisite ML models in a special-purpose engine or a multiuse

engine such as Spark. The data warehouse is an ideal platform for performing a workload of this kind, even if the RDBMS engine itself is not as efficient as an external, best-in-class ML-processing engine. After all, the data is already there. The requirements with respect to the engineering that needs to be applied in order to "push" this data "through" the ML models are well understood. The upshot is that it would be counterproductive to perform this workload in a separate engine—*provided* the data warehouse has enough capacity to accommodate it. In the cloud context, this is rarely a problem.

The caveat is that in-database processing of nonrelational workloads is in no sense a panacea. Take the ML-processing use case described above, for example. It is quite clear-cut, and its requirements—especially with respect to the shape of the data that it requires—are well understood. It is the finished product of extensive experimentation and training. But what if the reverse were the case? That is, what if the use case itself had not been clarified and the ML models had not been trained? In this event, the ML-processing engine would need to be able to (a) automate the selection and training of ML models and (b) parallelize this training workload across multiple compute nodes. The RDBMS engine at the core of the data warehouse is not designed for this. It would not "know" how to proceed.

This gets at something else, too: the multimodel capabilities of the data warehouse are valuable if the applicable use cases (and their requirements) are well understood. This is almost never the case in data science and ML/AI engineering, however; these disciplines are characteristically investigative, open-ended: they are efforts to *identify* use cases, to codify, engineer, and instantiate ML-driven, rule-based solutions that automate the actions (tasks, triggers, etc.) that are associated with these use cases. The multimodel data warehouse is less a useful tool for data scientific and ML/AI engineering than an ideal destination *for the products of this engineering*—that is, the thousands of ML models, for example, against which the large retailer (described above) processes its sales and demographic data.

The simple takeaway? The multimodel data warehouse gives organizations flexibility with respect to how and where they process certain types of workloads. It provides a convenient means to ingest, store, and query against graph, spatial, and time-series data. Commercial multimodel RDBMSs have long offered support for the spatial and time-series use cases; the availability of in-database graph capabilities is a relatively recent innovation, however. Commercial

vendors are keen to compare the capabilities of their multimodel RDBMSs against those of proprietary (graph-, spatial- and time-series-specific) systems. There is a reason, however, that markets exist for databases that are designed specifically for graph, spatial, and time-series functions: at a very low level, the engines that power these databases are optimized for operations involving a different kind of math (graph theory and/or linear algebra, as against the relational algebra that undergirds relational logic). If the RDBMS is a best-in-class platform for performing operations on relational data, a dedicated graph, spatial, or time-series engine will usually offer better performance than an engine that exposes similar capabilities.

CHAPTER 4

Migrating Your Data Warehouse to the Cloud

Migrating on-premises workloads to the data warehouse in the cloud is not just an opportunity to modernize data warehouse architecture, but an opportunity to transform the business. The migration process gives business and IT stakeholders, SMEs, and others a chance to talk about and clarify business objectives and priorities, and to brainstorm about the *new* applications—the new use cases and practices, yes, but also the new products and services—that are possible in the cloud.

The process of migrating an on-premises data warehouse to the cloud demands close collaboration between business and IT. The business must identify its most important stakeholders along with, if applicable, potential people or process problems. IT must do the same. The people-and-process dimension is a critical aspect of migration planning. Resistance on the part of internal constituencies is normal. In most cases, it is a function of fear, uncertainty, or doubt. It can often be resolved by involving resistant stakeholders in discussion and (to the extent possible) by working to address their concerns.

The actual migration of the data warehouse does not have to wait on any of this, however. This chapter discusses a few of the different strategies an organization can use to migrate its data warehouse system to a cloud context. The core points are (1) the process of migrating the data warehouse to a cloud environment not only

aligns with the goal of improving the data warehouse, but (2) is, ipso facto, an opportunity to reposition the warehouse for a new role and new use cases.

The Many Modalities of Cloud

SaaS, PaaS, IaaS: thanks to withering competition among cloud service providers, the terminology can get confusing. The good news is that this *is* the cloud we're talking about—in an age of commodity virtualization, no less! In practice, almost all providers offer *some* kind of a trial version that a would-be subscriber can either download and take for a test drive (usually as a VM) or sign up for free of charge. Now as ever, it behooves the would-be buyer to be skeptical—and to do their homework.

All this being said, it is not difficult to distinguish between the many modalities of cloud:

SaaS, or software-as-a-service
: The cloud provider manages *everything*: not just the virtual hardware layer, and not just the virtual software (database, middleware, operating system, etc.) that runs on top of it, but the applications that subscribers use and the data they generate.

PaaS, or platform-as-a-service
: The cloud provider manages the virtual hardware layer *and* most of the virtual software layer, with the exception of the applications subscribers build and the data they generate. PaaS exposes ease-of-use features that aim to mask the complexity of creating, configuring, populating, managing, and maintaining the warehouse. PaaS aims to be more or less turnkey—with the caveat that analytic development is *never* turnkey.

IaaS, or infrastructure-as-a-service
: IaaS is a scheme in which subscribers manage their own applications, data, databases and middleware, and operating systems. IaaS essentially replicates the features of on-premises data warehouse architecture in the cloud—albeit without subscribers assuming ownership of physical hardware. An enterprise that opts to migrate its on-premises data warehouse to IaaS should expect to maintain (e.g., apply security patches to) its virtual operating systems, databases, and middleware technologies.

Most cloud data warehouse vendors market *some* version of a PaaS data warehouse offering, and most on-premises vendors offer customers a choice between PaaS and IaaS versions of their data warehouse software. That said, the PaaS warehouse boasts a number of advantages relative to its IaaS kith—ease-of-use, ease-of-management, and simplified security foremost among them.

Cloud Data Warehouse Use Cases

If you are reading this book, you do not need to be convinced that you need a data warehouse. But you might be confused as to the role of the data warehouse in relation to the data lake—or the difference between the data warehouse and data marts, or the relation of so-called analytic sandboxes to the data warehouse (see Figure 4-1).

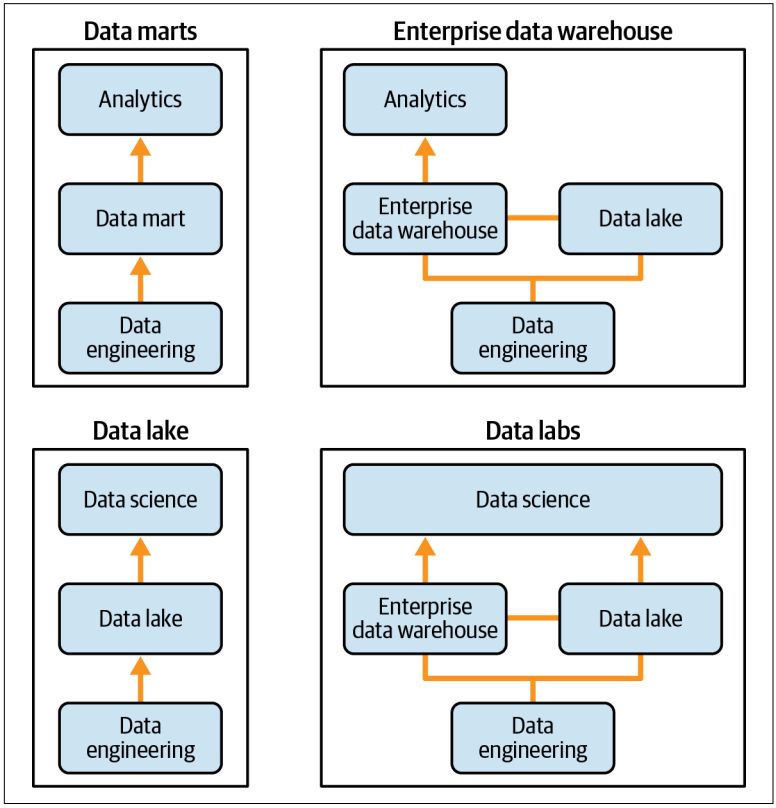

Figure 4-1. Modern data warehouse use case scenarios

This section explores each of these concepts with the idea of elucidating their strengths, limitations, and most appropriate use cases.

The Enterprise Data Warehouse

Why do we have a data warehouse? The short answer: we got sick of boardroom and conference room discussions devolving into arguments whenever two or more people disagreed about how, or where, they got the data to support their arguments. So we designed data warehouse architecture.

The data warehouse provides several services. These services are more or less the same in both the on-premises environment and the cloud, although it is worth emphasizing that the cloud context enlarges the role of the data warehouse. In both contexts, the data warehouse provides the following:

Operational reporting
: The role of the data warehouse does not change in the cloud. Here, too, it presents a single authoritative, panoptic view of useful business data. In the cloud, as in the on-premises environment, it permits decision makers to answer questions about the *whole* of the business, across all of its functional areas. It does this by ingesting data from different operational systems (finance, HR, sales and marketing, inventory management, etc.) and joining it together to create synthetic representations of the business subject areas that correspond to these operations. The operational reporting powered by the warehouse provides a kind of monitoring feedback loop into the organization and its operations.

  The data warehouse also provides an incisive analytic view into the performance of critical business operations. Organizations can define key performance indicators (KPI) for each business functional area and for the business as a whole that can be used to measure and to adjust its performance in real time. For example, financial KPIs might measure growth in revenue, net profit margin, cash flow, inventory turnover, and other indicators; sales KPIs, the net number of new sales contracts, the net number of qualified leads in the sales funnel, average time for conversion from lead to sale, and so on.

Ad hoc query

Not only do organizations need to monitor their business operations, but they expect to be able to identify and diagnose business disruptions. One function of the data warehouse's role as a feedback/monitoring loop is to alert decision makers to emergent problems. Another function is to permit decision makers to ask ad hoc questions about these problems: "What is going on here?" Quite aside from diagnosing problems, it is useful to be able to ask ad hoc questions about the business and its world. This is the function of the ad hoc query use case, which, on its surface, sounds like a simple enough problem: businesspeople need to ask questions that involve linking facts to dimensions in combinations that data modelers cannot possibly anticipate. In the background, answering these questions is technically and computationally demanding: the RDBMS engine that powers the data warehouse must join facts together with dozens or potentially hundreds of different dimensions in or close to real time. (Under the covers, the relational model itself enforces rigorous join logic, such that the joins the RDBMS performs are strictly valid based on the logic of relational set theory.) But ad hoc queries pose technical challenges for other reasons, not least because they can consume a disproportionate share of database resources, especially when multiple users query the data warehouse at the same time. This is why a viable data warehouse must support robust mixed workload and workload management capabilities.

History and planning

The data warehouse has a definitive historical dimension: the data in the organization's operational systems is always up to date, with fresh transactions overwriting stale data. But the data warehouse maintains a history of this data over time; this history permits decision makers to situate what is happening now, or what is projected to happen in the future, in a rich historical context. For example, is a seemingly anomalous event *really* anomalous? Is a business disruption anomalous? If not, what was its impact in the past? And what, if anything, could the organization have done differently to mitigate this impact? This combination of timely operational data and years of historical data supports not only day-to-day business decision making but long-term business forecasting and strategic planning.

Experimental data lab

Every business disruption or (seemingly) anomalous event is also an opportunity to test and to try something new. For example, are there indicators in the data coming from operational systems that the organization can use to predict a disruption or a (seemingly) anomalous event? Could data from *another* source be useful, especially in combination with operational data? If so, is it worth instantiating this insight into a (real-time) analytic view of some kind? This last example underscores an important point: the warehouse has always played a vital role as a kind of data lab for analytic experimentation.

This role changes in the cloud. In the on-premises data center, data warehouse capacity is comparatively scarce; for this reason, the data warehouse itself does not usually function as a lab or experimental sandbox. In the cloud, by contrast, warehouse capacity is not only practically limitless but easily created—and, moreover, just as easily destroyed. In the cloud, both humans and machines can rapidly instantiate a virtual data warehouse system to function as a sandbox for data processing and analysis or as a lab for interactive experimentation with data. We will say more about this in the section on ML and predictive analytics below. For the present, it is useful to note that virtual data warehouses can also be *triggered* in response to events (such as API calls) by both human and machine requestors.

ML, Predictive Analytics, and Other Emergent Use Cases

Some things about the data warehouse do change in the cloud—and, in most ways, for the better. In the on-premises data center, the data warehouse was sometimes starved for resources: organizations tended to run it at (or near) capacity, so it was difficult to accommodate all potential users, all potential use cases, and all potential practices at all times.

This changes in the cloud. Organizations can easily allocate new warehouse capacity—or create new, temporary virtual data warehouses—to accommodate the needs of BI discovery users, data scientists, ML engineers, and others. In the on-premises enterprise, too, data scientists and ML engineers tended to avoid the data warehouse. Again, it was difficult to obtain the resources they needed to do the work they wanted to do. The data warehouse was a stumbling block, not a potentially rich resource. This, too, changes in the

cloud. Some data warehouse systems already process ML functions in the database kernel itself; some likewise accept functional (Python) and imperative (R, Java, C) code as input, which makes them well-suited for data-scientific and ML workloads. In highly scalable data warehouse systems—which usually (but not always) are powered by a massively parallel processing (MPP) relational database—this can significantly accelerate ML processing. In the first place, it brings ML processing to the data itself, obviating the need to move data from the warehouse into a third-party engine for processing. Second, in-database ML functions also benefit from the inherent strengths of the MPP database engine: its native parallelism, its optimized arrangement of data structures in memory, its low-latencies, and its capacity for high-throughput read performance.

The *virtual* cloud data warehouse gives business analysts, data scientists, ML/AI engineers, and other skilled users a best-in-class context in which to prepare relational data to comport with the requirements of their work. After all, virtual data warehouses can easily be created and destroyed as part of event-driven workflows or data pipelines. In practice, a data scientist or a machine requestor can call an API that creates a new virtual data warehouse, which terminates when it is finished. When the data warehouse moves to the cloud, it stands to reason that a portion of data science and ML engineering workloads will gravitate back to it.

The Data Mart or Departmental Data Warehouse

The role of the data mart changes in the cloud, too. The data mart as such is a logical component of the data warehouse: that is, in data warehouse architecture, a data mart corresponds to a distinct business subject area. Historically, the *standalone* data mart (i.e., a data mart system instantiated outside of and separate from the data warehouse) has been the bête noire of data warehousing purists.

This has something to do with the history of the standalone data mart itself. Data warehouse architecture imposes strict constraints on the quality, conditioning, and use of the data that it manages. The on-premises environment imposes constraints of its own: the high cost of a data warehouse system; its limited storage capacity; and, last, its finite computational resources. The combination of these constraints functioned to limit access to warehouse resources for business analysts, BI discovery users, data scientists, and other

constituencies. So a business unit that was unable to get what it wanted from the data warehouse might create its *own* data mart, with its *own* data feeds. A business analyst who felt stymied in the same way might create her own data mart, and so on.

In the cloud, however, the standalone data mart does not necessarily need to exist. The role it occupied and the functions it fulfilled are better performed by the data warehouse itself. And some of the roles-of-convenience for which it was tapped—for example, as a subject-specific analytic sandbox—are better performed by the virtual data warehouse (i.e., a temporary—possibly respawning—analytic sandbox or R&D lab). In the on-premises enterprise, power users embraced data marts to perform work they could not do on the data warehouse. With respect to the cloud data warehouse, this changes. Some of these uses and use cases can now be accommodated in the context of the data warehouse itself—that is, with access to the data that lives in the warehouse. Others can be accommodated with a virtual warehouse sandbox.

The Data Lake

The data lake is home to useful data of all types and sizes, but it is not a replacement for the enterprise data warehouse. It cannot be held to the same strict standards with respect to the consistency and quality of the data that it stores. It cannot be expected to enforce consistent metadata definitions across its diverse subject areas—or "zones"—like the data warehouse.

It is conceivable that the data lake could be embedded in business processes, but it should not be interpenetrated with *business decision making*, as is the data warehouse. On a performance basis, its SQL interpreter is no match for the RDBMS engine at the core of data warehouse architecture. The combination of all of these factors makes it a poor choice for the SQL query and analytics use cases.

The primary role of the lake is as a practically unlimited *storage* layer—with an emphasis on storing data as distinct from managing it. A secondary role is as a platform for processing data and preparing it for use *in other contexts*, including the data warehouse. In these roles, the data lake supports a wide variety of users (human and machine alike) and use cases, from data scientists to data engineers, AI/ML engineers, software developers, and other skilled practitioners. These users tend to have radically different requirements,

especially with respect to the condition and quality of the data they expect to work with. The data lake usually exposes programmatic access to one or more data processing engines. The idea is that consolidating data in a single, central location makes it easier for different kinds of people—ETL developers, data scientists, data engineers, ML engineers, etc.—to access and manipulate data. Its built-in processing engines give them a means of testing and experimenting with this data *in situ*. They also simplify the task of preparing data for use in different contexts—including the data warehouse. The modern data lake usually incorporates ease-of-use and user-assist features to simplify certain kinds of repetitive or predictable use cases: for example, when the lake ingests a CSV file, a PDF document, and so forth, it indexes its contents and, if applicable, shreds numerical data into an appropriate storage format (e.g., attribute-value pairs, columnar data structures, etc.).

Data engineers, data scientists, and similar skilled technicians can optionally predefine their own data transformations and data extracts that get triggered under specific conditions. Think of the data lake as a community pool for different kinds of users, practices, and use cases.

The Cloud Data Management Stack

As discussed earlier (see "Modern Data Warehouse" on page 15) the cloud data warehouse is complemented by a data lake, which is also situated in the cloud. The two contexts comprise a kind of logical, if federated, whole: the logical data warehouse. Some organizations will deploy a cloud data warehouse that is colocal with—that is, hosted in the same cloud infrastructure service as—their data lake. This simplifies the process of moving useful data between the two environments and making it available for use.

In practice, both the warehouse and the data lake are used for some of the same tasks: for example, storing data, accessing and retrieving it, processing and transforming it, and performing analytic operations on it.

It is worth exploring how these tasks are managed differently in each environment:

Data access and retrieval
: The modern data lake ingests data of all types and sizes. It exposes SQL, REST, and one or more language specific libraries—for Python, R, Java, and so on—that human users (data scientists, ML engineers, and data engineers) and machines can exploit to access and manipulate data. It generates metadata and—concomitant with the ingest, access/manipulation, and movement of data—keeps track of data lineage, too.

  What is more, the data lake is a **superior alternative** to the data warehouse for storing multistructured data *in its original format*, as well as for performing operations on multistructured data and on certain kinds of semistructured data, such as nonplain-text documents.

  But the data lake is an **inferior option** for the SQL query and SQL analytics use cases. On average, the SQL interface exposed by the data lake supports only a subset of the ANSI SQL standard; few (if any) relational databases support all ANSI SQL revisions, but most implement complete support for older ANSI SQL revisions (e.g., through ANSI SQL 1999) and less consistent support for more recent revisions. Some RDBMSs also incorporate proprietary SQL extensions that simplify or accelerate certain kinds of queries or analytical operations.

  The upshot is that the RDBMS is and will remain a best-in-class engine for SQL query and for virtually all types of analytical operations on relational data.

Data engineering and transformation
: Because data gets vectored into the data lake from all points in the enterprise's world, it has emerged as the locus of data ingestion and engineering. For most organizations, the site of the lake has shifted from the on-premises data center to the public cloud. This fact has especial salience when the data warehouse is *also* hosted in the public cloud. In this scheme, the lake can in

effect become the ETL layer,[1] ingesting data from operational systems (many of which have also moved to cloud) and used by ETL developers to cleanse and transform the data destined to populate the warehouse. The advantage of doing this is that organizations can retain rich detail data that would otherwise be discarded during the ETL process. This data is useful grist for data scientists, ML/AI engineers, and other users.

The on-premises data warehouse will continue to depend on its ETL layer for data access. It will draw data directly from on-premises systems—so long as these systems remain in the data center. But it, too, now derives a large (and growing) share of its data from the lake. In these and other ways, the lake has become a critical feeder tributary for the warehouse.

The data lake feeds not only the data warehouse but a wide range of different analytic practices and use cases. It complements the data warehouse in other types of practices and use cases: hosting analytic sandboxes and data labs. But the emphases in these data-lake-focused analytic environments is typically on engineering data and preparing it for use in another context: data scientists, ML engineers, and similar users typically create data pipelines (see Figure 4-2) in interactive analytic notebooks to schedule operations on data and to move data between repositories and data processing engines. These notebooks can exploit the data lake's built-in processing capabilities, along with those of the data warehouse—to say nothing of a medley of cloud data processing engines or services, too.

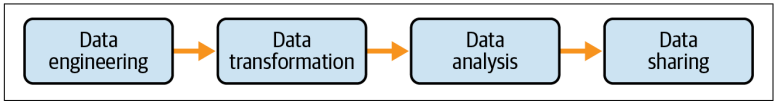

Figure 4-2. Data orchestration/data pipeline

1 For all practical purposes, the focus of ETL itself has already shifted to cloud—or, more precisely, to the cloud data lake. Most commercial ETL vendors also market cloud offerings that support more than one public cloud provider and specific cloud SaaS and PaaS offerings. Commercial ETL vendors specifically target the cloud data lake use case, too, typically via provider-specific offerings. And most public cloud providers offer an array of data integration services suitable for different kinds of use cases or for users in a breadth of roles or practices—from software developers to ETL developers and architects to self-service users of different types.

Contextualizing Data in the Cloud

Data lives in the data warehouse, where everything is known about it. It lives in the data lake, where less is known about it. But it likewise lives in sandboxes and other repositories dispersed throughout the on-premises enterprise and the cloud. It lives in cloud storage services.

The remit of the data warehouse is to provide a unified—panoptic, as this book puts it—view of useful business data. But the fundamentally federated logistics of the cloud complicate this. Some cloud data warehouse systems use technologies such as data virtualization and data catalog services to simplify the tasks of discovering, labeling, and exposing useful data. Others incorporate useful capabilities —such as graph-processing—in their RDBMS engines. Collectively, these technologies aim to make it easier for subscribers to knit together data scattered across both on-premises environments and the cloud.

Data Virtualization

Formerly known as "data federation," data virtualization is used to facilitate a single view of data across all cloud *and* on-premises contexts. Data virtualization exposes a logical, unified view of data that, in the physical world, is cobbled together from disparate, geographically distributed data sources. A decade ago, data virtualization was commonly used to create canonical "business" and "application" views, which could be thought of as analogous to virtual data marts. In the era of the data lake—or of multiple data *lakes*—data virtualization is used to present canonical views of semistructured and polystructured data, too. It is a proven, useful technology for knitting together disparate data sources.

With respect to the data warehousing use case, data virtualization is much less useful for ad hoc query than for building models (views) that support common or recurring queries. Because data virtualization distributes queries across multiple (often geographically distributed) contexts, it gives priority to minimizing data movement: when possible, its engine pushes data processing up to source systems. (Over time, data is also cached in the data virtualization tier.) Data virtualization exposes a single interface for SQL query; under the covers, however, it uses smart technology to accelerate query

performance. Several cloud data warehouse providers offer a virtualization layer of one kind or another.

Data Catalogs

Data catalogs permit users to discover, profile, and classify data, regardless of context. Even though data catalog technologies are typically deployed and managed by IT, they are *used* by self-service power users: for example, BI discovery users, business analysts, data scientists, ML engineers. Data catalog technologies help users discover, label, and procure useful data for analysis. More advanced data catalog services permit users to manipulate data, even automating the profiling and transformation of data for analysis. These services also incorporate rich metadata management features, along with (often less rich) capabilities for tracking and capturing data lineage.

Data catalogs are frequently used in conjunction with data virtualization: an analyst, BI discoverer, or other skilled person discovers useful data in one or more contexts—a cloud data warehouse service; a zone in a data lake; a cloud storage service—and works with IT, business subject-matter experts, and data modelers to expose it via data virtualization. Several PaaS vendors offer more or less useful versions of these technologies. (In the IaaS cloud, data virtualization and data cataloging are either included with the core data warehouse or licensed separately.) This usefulness is not specific to just the data warehouse: they knit together data that is scattered among disparate contexts.

Graph Databases

The slice of the business world that is captured by traditional BI analytics is relatively narrow.

It ignores the data that lives in time-series databases, hierarchical data stores, network databases, document databases (and similar content management systems), to say nothing of the data lake. It ignores the wild profusion of cloud data sources. But deriving context (i.e., linking text-analytic data with time-series data with attribute value-pair data with relational data) is a hard problem. It involves using a technique known as graph traversal, which is the remit of so-called graph databases.

This is why some data warehouse platforms incorporate graph processing capabilities. The beauty of embedding a graph engine in (or along with) an RDBMS engine is simple: you can actually store graph data in the RDBMS, then use SQL to access and query the graph data: the database's built-in graph-processing engine automates the task of translating SQL queries into establishing context.

CHAPTER 5

The Data Warehouse in the Cloud

For the purposes of on-premises-to-cloud data warehouse migration, most organizations tend to deploy in accordance with one of two topologies. The first is a so-called hybrid cloud—that is, one that knits together an on-premises data warehouse with a data warehouse in the cloud. The second is a so-called *multicloud* topology, which distributes the cloud data warehouse across two or more providers. A few variations on these themes (such as hybrid-multicloud) are common, too.

Cloud Data Warehouse Topologies

Let's briefly review these topologies.

Hybrid Cloud

A *hybrid deployment* is a data warehouse that spans two distinct contexts—typically, the on-premises enterprise and the cloud. For our purposes, there are two kinds of hybrid configurations:

On-premises nonvirtualized data warehouse <> cloud data warehouse
: This is a relatively common scenario. One example of this is when an organization migrates its BC/DR, test-dev, and analytic discovery workloads to the cloud data warehouse and keeps its production workloads in a nonvirtualized on-premises system.

On-premises virtualized data warehouse <> cloud data warehouse
: This is less common but becoming more so—especially now that most PaaS providers also support on-premises private

cloud deployments. Hybrid deployments of this type might focus on hosting certain kinds of workloads (e.g., those that involve sensitive data, those that are especially demanding) in the on-premises virtual data warehouse.

Same-vendor to same-vendor hybrid cloud deployments can be advantageous for several reasons, not least because of portability between cloud contexts: SQL, indexes, procedural code, UDFs, database-specific schemas, and similar assets will often (but not always) transfer without issue from one context to another. Most database-specific skills should transfer, too. For these reasons, providers like to tout an ability to shift workloads between contexts in a homogeneous hybrid cloud.

Multicloud

Multicloud means just what it sounds like: rather than sourcing its cloud services from a single provider, the organization distributes its data warehouse across two or more cloud providers. It is necessary to distinguish between a multicloud *strategy* and *tactical* use of multicloud.

Multicloud in its strategic dimension can mix elements of risk management—namely, an emphasis on hedging against the risk of service provider lock-in—with an à la carte shopping experience (different cloud services have different strengths) and with BC/DR planning, too.

In practice, a mix of cloud and on-premises data warehouse systems is not uncommon. This is less an example of a hybrid-multicloud deployment—in which an on-premises data warehouse and two or more cloud systems coexist *inter pares*—than a hybrid deployment in which the organization makes tactical use of at least one other cloud service. So, for example, an enterprise might host test-dev and BC/DR in one cloud service (usually identical to that of its on-premises provider) and one or more analytic sandboxes (used to support its business analysts) in another. Some enterprises might host their data lake and their data warehouse in one cloud but support several different business use cases in sandboxes in another cloud. Some cloud providers charge for capacity, some for use, some for both.

The Local Cloud, the On-Premises Public-Private Cloud

The PaaS data warehouse is not strictly confined to the distant public cloud. In just the last two years, several vendors have introduced services that bring the PaaS data warehouse closer to—and, with respect to the on-premises public-private cloud, *directly to*—the enterprise. One strategy is to bring public cloud infrastructure to the enterprise edge: several hyperscale providers operate local "zones" in metropolitan areas that are not otherwise adjacent to one of their large regional hosting centers. These local zones bring cloud infrastructure—compute, storage, and network resources, as well as SaaS and PaaS services that run atop them—closer to the enterprise. Another strategy, the on-premises public-private cloud, consists of a PaaS data warehouse that is installed in the customer data center by the cloud provider. The provider owns and maintains the hardware and the supporting infrastructure software (including the functions of the PaaS data warehouse that are abstracted from the customer) and the customer usually pays for power, cooling, and other data center essentials.

These services, and the public-private PaaS cloud in particular, address several problems. In some cases, data and workloads just cannot leave the on-premises environment. Some customers (particularly in government) are constrained by statutory or regulatory requirements that prevent them from moving data to the public cloud. Some are just uncomfortable moving sensitive data to an off-premises context, be it the public cloud or a regional hosting facility. Some have contracts with customers, partners, suppliers, etc., that prevent them from moving certain kinds of data off-site.

Finally, and no less important, some organizations may be unable to move demanding workloads (and the data associated with them) because of the performance limitations of public cloud infrastructure.

This last is usually a function of the suboptimal latency of public cloud infrastructure. In the topology of the public cloud, virtual (that is, software-defined) resources are typically *nonlocal* with one another; in most cases, this means that virtual resources communicate with one another over network transport, not via high-speed local computer buses. In the public cloud, the messages and data that operating systems, databases, and middleware exchange with one another—which, in an on-premises, nonvirtualized SMP data

warehouse, would transit a high-speed local bus—use this network transport, too. (In the on-premises data center, MPP systems tend to use some form of high-speed, low-latency interconnect technology, such as InfiniBand, to exchange data and messages between nodes. Today, several cloud-providers offer low-latency cloud data warehouse services that run atop InfiniBand.)

The upshot is that—in most cloud PaaS data warehouse topologies—the virtual "disks" that the data warehouse reads data from and writes data to are actually implemented in *object storage*, a kind of distributed file system that spans hundreds or thousands of disk drives. Although object storage has its benefits—it supports high-data-throughput sequential reads and writes—it tends to have very high latency. As a general rule, the speed with which an analytic workload can be processed is inversely related to latency: the lower the latency, the faster the speed at which the workload can be processed.

Bringing public cloud infrastructure to a local or metropolitan zone helps reduce latency. For example, in a large regional hosting center, virtual storage resources do not live in the same rooms—sometimes not even in the same *buildings*—as the virtual servers that connect to them. In the local zone, by contrast, resources are closer together (ideally, in the same data center facility), which reduces latency and helps make performance much more predictable. The on-premises public-private cloud goes this one better, however, by consolidating compute and storage into a single, integrated system. Data and messaging traffic still transit over software-defined links, but because the underlying compute and storage hardware lives in the same cabinet (or in one or more adjacent cabinets), performance is comparable to that of a conventional data warehouse system. For especially demanding workloads, or for data warehouse systems that host hundreds of concurrent users (and workloads) of different types, the on-premises public-private cloud offers the lowest latency and best possible performance.

Selecting a Provider

An organization that has a large on-premises data warehouse system will tread carefully as it migrates this system to a cloud service. "Large" in this context means the data warehouse hosts a mix of different workloads, supports a large number of concurrent users,

and/or is the focus of diverse practices—that is, in addition to its core role of supporting BI reporting and analytics.

The key recommendations for pursuing strategy are:

- Protect IT investments that have already been made over the years
- Migrate to the cloud without disrupting the business
- Extend the existing environment by adding new innovations and technologies

An organization with less demanding requirements has more flexibility in selecting a cloud provider.

In all cases, the organization's choices are determined by its priorities. For example:

1. Is the organization happy with its existing (on-premises) data warehouse vendor?
2. Relatedly, does the organization's on-premises vendor market its own PaaS data warehouse?
3. Relatedly, does the organization expect to maintain a hybrid (on-premises + cloud) topology?
4. Where do the organization's operational applications live? Are most of them still in the on-premises environment, or have some/all also shifted to the cloud?
5. Relatedly, is intracloud integration with the organization's cloud infrastructure a requirement?
6. Are there needs—SLAs, high concurrency, support for mixed workloads, compliance with security or other standards—that only a few providers can realistically meet?
7. Are there esoteric analytic needs (e.g., in-database graph, ML, time-series, etc., processing)?
8. Is integration with other cloud services—not only cloud SaaS applications but also cloud ML, data integration, AI, software development, etc., services—a priority? Should it be?

These are just a few of the questions that factor into the calculus of choosing a cloud provider.

An organization with specialized needs should be able to quickly narrow down its list of providers.

There are a few points worth emphasizing, however:

1. PaaS data warehouse services are not interchangeable. To adapt from Leo Tolstoy, each PaaS data warehouse is unalike in its own way. Some PaaS data warehouses have superior mixed-workload capabilities, which means they can host a large number of workloads simultaneously. Some incorporate useful features (e.g., in-database ML, graph-, or time-series-processing capabilities) that set them apart. Others have superior cloud-specific feature sets. They spin up and shut down faster, or they pause and resume more quickly. They have friendly, more intuitive management and design tools.
2. The tactical use of more than one cloud data warehouse service is not strictly inimical to the goal of having a unified view of useful business data. You have a data warehouse: it is the authoritative repository for all useful business data. It supports business planning and decision making. It provides a panoptic view of all useful data. But you have analytic sandboxes and R&D data labs, too. If necessary, you use data virtualization, data catalog services, and other technologies to knit these resources together. This is the remit of the logical data warehouse, discussed in Chapter 2.
3. The tidal pull of operational applications is strong. This force is especially strong once on-premises applications move to the cloud. Several factors—including the asymmetrical cost of moving data out of the cloud; security; simplified development—amplify this tidal force. The pressure to host the data warehouse in an intracloud context with operational applications can prove to be decisive.

Configuring and Managing the Data Warehouse in the Cloud

In the on-premises data center, the sizing and optimization of the data warehouse is a focus of considerable attention. DBAs and other skilled technicians expend considerable time and effort designing, configuring, managing, maintaining, and (of course) *troubleshooting* the on-premises data warehouse. How does this change in the

cloud? Mostly for the better. Consider a few of the choices and responsibilities that are bound up with the implementation and use of an on-premises warehouse:

Choosing a conventional or MPP data warehouse system
: Many cloud providers offer MPP or MPP-like[1] data warehouse services. An MPP database is the best overall option for demanding analytical requirements. The brawn of the MPP data warehouse offsets many of the performance constraints of the cloud model. Not only is it cost-effectively priced, but cloud software—in the PaaS model, at least—helps automate the creation, configuration, deployment, and maintenance of an MPP database.

Configuring and sizing the data warehouse system
: This, too, is more challenging with respect to conventional, as distinct from MPP, database systems. A conventional data warehouse is instantiated on a single physical server, which means DBAs and other technicians must anticipate maximum (and future) needs as they design the size and capacity of the data warehouse system. An MPP system is somewhat easier to size, inasmuch as enlarging the capacity of the MPP cluster is a function of adding new server nodes. But the on-premises MPP warehouse is constrained by the hard limits of cost—large MPP databases are expensive!—and physical capacity; the cloud MPP warehouse is not.

Scaling and maintaining the data warehouse system
: In the on-premises data center, scaling a conventional data warehouse system is more challenging than scaling an MPP data warehouse system. There are hard limits as to how many processors and how much memory can be stuffed into a single server. (Scaling an on-premises MPP system is in no way simple, however.) In the PaaS cloud, software functions automate the creation and much of the management of the virtual data warehouse system. Software also automates the expansion of the virtual data warehouse system, along with its balancing and

1 Some cloud services achieve something *like* an MPP topology by employing a shared storage substrate and independent compute nodes. That is, all compute nodes share access to all of the tables in the database, but each processes workloads independently of the others. This scheme achieves MPP-like compute performance.

optimization. In an even more basic sense, software automates the sizing up or sizing down of the virtual data warehouse. These are nontrivial tasks in the on-premises enterprise.

Troubleshooting the data warehouse system
: DBAs and developers expend a nontrivial amount of time troubleshooting performance issues with the on-premises data warehouse. This is true even in the case of MPP data warehouse systems. The data warehouse in the cloud upends this status quo: for one thing, the service provider is responsible for troubleshooting hardware bottlenecks. "Fixing" these issues is easier in the cloud, too: service providers can architect around problems by allocating additional resources. In the on-premises data center, DBAs do not have this flexibility.

At some point, subscribers will encounter problems that can be fixed by neither the abundant resources of the cloud nor the automation and intelligence built into even the most sophisticated of cloud infrastructure platforms. In the cloud data warehouse, as in its on-premises predecessor, query performance is as much a function of software smarts (the core database's query optimizer) as of hardware brawn. Some database query optimizers are better than others. Some databases offer better features—more insight and granularity with respect to the query optimizer's query plan and its impact on system resources—than others. DBAs, developers, and other technicians must still use their smarts to troubleshoot these and similar issues. They should have more time in which to do so, however.

Securing the Data Warehouse in the Cloud

All things being equal, the cloud data warehouse is at least as secure (and probably *more* secure) than the on-premises data warehouse. Cloud service providers tout compliance with industry-standard guidelines (as defined by the Cloud Security Alliance, NIST, FedRAMP, PCI DSS, and other relevant specifications) and employ commonsense best practices, such as enforcing password length/aging requirements; encrypting data by default; separating sensitive from nonsensitive data; and supporting multifactor authentication. Cloud service providers perform security tests (audits, penetration testing, scanning for vulnerabilities, etc.) on a more frequent and consistent basis than IT teams in the on-premises enterprise.

Security is a net positive for the cloud as against the on-premises environment.

Security in the PaaS cloud, especially, constitutes a reprieve for DBAs, system administrators, and others. In the on-premises environment, enterprises are solely responsible for ensuring the integrity and consistency of their data. They need to be careful about how and under what circumstances they store sensitive data, both to safeguard against possible data breaches and to comply with pertinent data privacy laws. Relatedly, they have a responsibility to comply with applicable data movement, data retention, and data deletion requirements, some of which are region-specific. Lastly, they need to be able to protect their *physical* assets—not just server, storage, and network resources, but *business campuses* (with their buildings, doors, windows, etc.), too. In other words, an enterprise that shifts its applications or workloads to the PaaS cloud shifts a significant proportion of security-related risk, too.[2]

Another problem has to do with sensitive data. In the cloud, a common best practice is to separate sensitive from nonsensitive data, so most cloud service providers expose wizards (or quasi-automated features) that aim to make it easier to create rules for managing sensitive data. Some also use ML models to profile data during ingest, which helps automate the identification of sensitive data. Managing sensitive data poses an additional difficulty with respect to data warehousing, however: analytics are constructed by combining sensitive with nonsensitive data. So some providers use automated technologies (such as data hashing, anonymization, or differential privacy) to strip data of sensitive details. The masked data is still useful for analysis but will not leak or disclose information.

A final, related consideration is that almost all PaaS data warehouse services turn data encryption *on* by default. This is not strictly true in the on-premises enterprise, however; and in the IaaS data warehouse, too, data encryption is usually the subscriber's responsibility. By itself, this presumption of data encryption is an advantage for the PaaS data warehouse. But the PaaS model helps simplify the *logistics* of data encryption, too. For example, one of the most unnerving

2 In most IaaS data warehouse services, as in the on-premises enterprise, the subscriber owns the responsibility of maintaining virtual operating system, database, and middleware software.

aspects of encrypting data has to do with the problem of managing encryption keys. In the PaaS data warehouse, the hosting provider assumes responsibility for managing encryption keys, ensuring that key management services are online and available, that new keys get distributed to all relevant databases, and so on.

Conclusion

The data warehouse in the cloud does not constitute a radical break with its on-premises predecessor. In the cloud, the role of the data warehouse is the same as it ever was: it remains *the* authoritative system of record, *the* ground and guarantor of the veracity of all of the data that is potential grist for business decision making. Even prior to the emergence of data lakes and data science, one critical role of the data warehouse was that of a research lab in which *the business performed experiments on itself*.

The data-warehouse-in-the-cloud, by contrast, is practically unlimited in terms of its size and prolificacy. It is *multiparous*—that is, capable of being recreated or replicated as often as needed—in much the same way that the on-premises data warehouse is not. Subscribers can draw upon the reserve capacity of the hyperscale cloud to create very large single-instance data warehouse configurations of dozens or, even, hundreds of terabytes. They can create, pause, resume, and/or destroy virtual data warehouse instances as needed; better still, instances can be created (or destroyed) in response to programmatic events, such as API calls, or triggered by rules engines. *The data warehouse in the cloud is not perfect.* As a general rule, on-premises data warehouse systems will require *more* compute, *more* memory, and *more* storage resources if they are to be successfully transplanted into the cloud context. *How much more* is a function of trial and error.

The cloud data warehouse of today is more performant than that of, say, half a decade ago. A more recent innovation is that of the on-premises public cloud: a PaaS service that is either deployed in a local hosting "zone" (i.e., a local data center) or in the customer's

own data center. In this scenario, the cloud provider, rather than the customer, owns and manages the service. Cloud-at-customer gives subscribers another (albeit premium) option to address stubborn performance issues. This has everything to do with the inexorable economics of cloud. It is not just a force for driving down costs but for doing so while also improving performance.

In the cloud context, a data scientist can easily design data pipelines that (1) extract useful data from the production data warehouse, be it on- or off-premises; (2) call RESTful APIs to spin up one or more virtual data warehouse instances; (3) load the extracted data into a virtual data warehouse instance; (4) perform one or more specific operations on this data; (5) integrate it with data prepared in another context; (6) move the resulting dataset to a destination repository; and, lastly, (7) destroy the virtual data warehouse instance. This scenario is *possible* in the on-premises enterprise, to be sure; however, owing to physical, economic, and logistical constraints, it is difficult to implement in practice.

Extending the data warehouse to the cloud helps simplify disaster recovery/business continuity (DR/BC) planning. Shifting DR/BC from the on-premises data center (or from a leased DR space) to the cloud is one of the most popular migration scenarios: low-hanging fruit, as it were. But the data warehouse in the cloud also transforms an organization's security posture. Cloud service providers—and PaaS providers, especially—tend to be better about enforcing commonsense security safeguards, such as on-by-default data encryption, or password-complexity and aging requirements. All things being equal, the data warehouse in the cloud is a more secure platform than its on-premises kith.

Because the data warehouse in the cloud is still a site of rapid transformation, organizations should not expect to move all of their on-premises workloads to the cloud. Right now, it may be neither cost-effective nor possible to do so. In the overwhelming majority of cases, subscribers will need to allocate additional resources to achieve performance that is at parity with their on-premises data warehouse systems.

The economics of cloud make data warehousing relatively cost-effective, but not all cloud data warehouses are alike. Some are more adept at scaling and managing challenging workloads, as in the case of a data warehouse system that hosts a mix of workloads of

different kinds, or a large number of concurrent users. Some cloud data warehouses are "cloudier"—*more* elastic; *more* flexible; quicker to start, pause, and resume—than others. These factors also impact performance and availability.

An organization should thoroughly test-drive a cloud data warehouse service prior to inking a contract with a provider. Organizations that have large data warehouse systems should aim to test against *a complete copy of their data*, if possible; they should also evaluate other factors, such as data loading performance, the reliability and performance of their ETL jobs, even the speed of database inserts. If applicable, they should attempt to account for the cost of workloads that involve data egress to other cloud services—or to on-premises resources. But due diligence of this kind will save adopters time, money, and frustration in the long run.

Final Thoughts

The data warehouse in the cloud is transformative in another important way. Depending on the provider, the cloud data warehouse is logically adjacent to (or "lives" in the same context as) cloud-based ML, AI, data integration, and developer-oriented services. These services are already quite popular—not only with ML and AI engineers but with software developers, too. In its most recent *Magic Quadrant for Cloud AI Services* report,[1] for example, market-watcher Gartner projected that—by 2023—40% of development teams will use AI services to incorporate AI capabilities into their apps. Five years from now, *half* of all data science activities will be automated by AI, according to Gartner.

Today, this is a vision, especially when the data warehouse remains tethered to its on-premises launch pad. But transplanting the data warehouse from the on-premises data center into the cloud is an important, a conative, first step. The cloud data warehouse is a new home for old workloads, yes; more important, it is a site for and a focus of new kinds of workloads and new types of analytic development, allowing you to maximize the value from data, understand it better, and drive efficiencies and innovation.

1 *Magic Quadrant for Cloud AI Services* (Gartner, February 24, 2020).

About the Author

Steve Swoyer is a writer, researcher, and analyst with more than 20 years of experience. His research focuses on business intelligence, data warehousing, and analytics, as well as edge issues in data science, machine learning, and artificial intelligence. Steve enjoys researching and writing about emerging trends and potentially transformative ideas in systems design and architecture. As an analyst and researcher, Steve explores trends in distributed systems architecture, cloud native architecture, and other emergent subject areas. He is a recovering philosopher with an abiding interest in ethics, philosophy of science, and the history of ideas.